Dual Sentinel: Intrusion Detection and Prevention System

Mansij Pandey¹, Yogita Nere², Ankita Khandare³, Priyanka Singh⁴

¹²³⁴Dept. of computer, NDMVP’s KBT COE, University of Pune

Abstract—Now a days, the use of internet has been increased so rapidly and on large scale which has increased the number of applications and complexity of data. In order to deal with this increase in application and data complexity, web services are designed as multitier services. In this paper, we present Dual Sentinel which is used to detect attacks in multitier web application. By monitoring both web and consequent database requests, we are capable of searching out attacks that an independent ID would not be able to. This approach can create normality models of isolated user sessions that include both the web front end and back end (File, Sql). We implemented Dual sentinel using an Apache server with MySQL and virtualization. We then gathered and developed real-world traffic over a 15-day period of system exploitation in both dynamic and static web applications. Finally, using Dual sentinel, we are able to expose an extensive range of attacks with 100% correctness still maintaining 0% false positives for static web services and 0.6% false positives for dynamic web services.

Keywords—multitier web applications, intrusion detection, virtualization and anomaly.

I. INTRODUCTION

At the present time, web-delivered services such as online banking, online shopping, social networking, admission process etc. have become enormously admired as well as extremely complex. These services particularly utilize a web-server.

Web server front-end which runs the application user interface logic and a back-end database server that consists of a database or file server. As these services are used everywhere, they are highly susceptible to attacks. These attacks have recently become more miscellaneous, as concentration has shifted from attacking the front end to exploiting vulnerabilities of the web applications, to corrupt the back-end database system (e.g., SQL injection attack). An overabundance of Intrusion detection systems (Intrusion Detection Systems) currently examines network packets individually within both the web-server and the database system. However, in multitier architectures, the back-end database server is frequently confined behind a firewall while the web-servers are distantly accessible over the Internet.

Also, we used Zero Knowledge Protocol which allows identification, key substitute and other such kind of fundamental cryptographic operations to be implemented without enlightening any secret information during the conversation and with smaller computational requirements. Zero Knowledge Protocol prevents certain active attacks like, Man-In-The-Middle-Attack or Replay Attacks.

II. RELATED WORK / LITERATURE SURVEY

A network Intrusion detection system (Intrusion Detection System) is mainly categorized into the two types: anomaly detection and misuse detection. In Anomaly detection, the accurate and acceptable static form and dynamic behaviour of the system is defined first. And this is being used to identify the changes or abnormal behaviours. Then an anomaly detector compares current patterns with the models that are already well-known so as to recognize anomalous events. We follow the anomaly detection approach because we are dependent on a training phase to build the correct model.

Dual sentinel uses the container ID for each session to causally map the related events, whether they are concurrent or not. Databases should be given highest level of security as they contain crucial information. The system is made up of both Web Intrusion Detection System and database Intrusion Detection System so as to accomplish more precise detection. It makes use of a reverse HTTP proxy for maintaining a condensed level of service when fake positives are present.

However there exist certain types of attacks which can make use of regular traffics and hence they cannot be detected by web Intrusion Detection System and database Intrusion Detection System. So we used Virtualization with the intention of separating objects and improving security performance. A lightweight virtualization, such as OpenVZ, Parallels Virtuozzo, or Linux-VServer is some of the alternatives. In our Dual sentinel, we utilized the container ID so that we would be able to separate session traffic for identifying causal relationships between web server requests and database query events.

We can initialize thousands of containers on a single physical machine. And those virtualized containers can be discarded, reverted, or rapidly reinitialized for allocation of new sessions.
An Intrusion Detection System also uses temporal information to detect intrusions. Dual sentinel, however, does not correlate events on a time criteria, which runs the risk of erroneously considering independent but concurrent events as correlated events. Dual sentinel does not have such a limitation as it uses the container ID for each session to causally map the related events, whether they are seem to be concurrent or noncurrent.

As a single physical web server can run many containers, each of them is an exact carbon copy or replica of the original web server. We follow an approach which can not only dynamically generate new containers but also recycles used ones. Hence a single physical server can run continuously while serving all the web requests. Each session is logically separated from other session. We initialize each virtualized container by using a read-only clean template. We guarantee that each session will be served with a clean web server instance at initialization.

### III. Threat Model and System Architecture

We firstly set up the threat model so that to include our assumptions and the various types of attacks we are supposed to protect against using dual sentinel. We assume that both the web and the database servers are vulnerable. The attacks are network driven and they come from the web clients; they can launch application layer attacks to compromise the web servers they are connecting to. Attackers can avoid the web server and can directly attack the database server. We have assumed that the network attacks can neither be detected nor prevented by the current web server Intrusion Detection System, that attacker may take over the web server after the attack, and then they can obtain full control of the web server to launch successive attacks. For example, the attacker can modify the application logic of the web applications, and also eavesdrop or may also hijack web requests of other users, or modify the database queries to steal sensitive data beyond their privileges. On the other hand, at the database end, we have assumed that the database server will not be completely taken over by the attackers.

We also assume that no attack would occur during the training phase and model building.

Attackers may strike the database server through the web server by submitting SQL queries, they may corrupt susceptible data within the database. We assume no prior knowledge of the source code or the application logic of web services deployed on the webservers.

In addition, we are analysing only network traffic that reaches the web server and database.

### IV. Overall Description

Dual Sentinel is a system used to detect attacks in multitier web services. This System is able to create normality models of isolated user sessions that include both the web Front-end (HTTP) and Back-end (File or SQL) network transaction. In Dual Sentinel the new container based Web Server architecture enables to separate the different information flows by each session. This can provide a means of tracking the information flow from the web server to the database server for each session.

State Chart Diagram:

- S_0=Request manager
- S_1=Authenticator
- S_2=Container manager
- S_3=Query analyser
- S_4=Alert Generator
- S_5=Blocker
- S_6=Database

\[\text{Fig. 1: state chart diagram}\]

### V. Classic Three-Tier Model

Now we will see the classic three-tier architecture model. We are unable to tell which transaction corresponds to which client request at the database side. The communication between the web server and the database server is not separated, and we can understand the relationships between clients and server.
We have created multiple sessions in that, each and every session assigned to a dedicated web server and separated from other sessions. So at the initial stage web server served as a clean template by using virtualized container.

VI. BUILDING NORMALITY MODEL

This container-based and session-separated web server architecture enhances the security performances and also provides us with the inaccessible information flows that are separated in each container session. Hence it allows us to spot the mapping between the web server requests and the succeeding database queries, and can utilize such mapping model to detect abnormal behaviours on a session level or client level.

In typical three-tiered web server architecture, web server receives HTTP requests from users and then issues SQL queries to the database server to retrieve and modify data. These SQL queries are fundamentally dependent on the web request hitting the web server. We want to model such fundamental mapping relationships of all legitimate traffic so as to detect abnormal traffic. In practice, we are cannot build such mapping under a classic three-tier setup.

Although the web server can distinguish sessions from different clients or users, the SQL queries are mixed and they are all from the same web server. Hence it is impossible for a database server to decide which SQL queries are the results of which web requests, and much less to discover out the relationship between them. Though we knew the application logic of the web server and were to build a correct model, it would be highly impossible to use such a model to detect attacks within huge amounts of concurrent real traffic unless we had a mechanism to identify the pair of the HTTP request and SQL queries that are fundamentally produced by the HTTP request. Still, within our container-based web servers, it is an uncomplicated matter to identify the causal pairs of web requests and resulting SQL queries in a given session. Furthermore, as traffic can easily be separated by session, it has made it possible for us to compare and analyse the request and queries across different sessions.

To that end, we put sensors at both sides of the servers. At the web server, our sensors are installed on the host system and cannot be attacked directly since only the virtualized containers are exposed to attackers. These sensors will never be attacked at the database server also, as we have assumed that the attacker is not able to completely take control of the database server. Actually, we have made an assumption that our sensors cannot be attacked and can always capture correct traffic information at both ends. Above figure shows the locations of our sensors.

Once we build the mapping model, it can be used to detect abnormal behaviours. Both the web request and the database queries within each session should be in accordance with the model. If there exists any query or request that violates the normality model within a session, then that session will be treated as a probable attack.

Algorithm:

1. User enters id and password.
2. Authentication
   If (successful) then
   Send web request
   Else
   Renter id and password
3. Request checking
   If (safe request) then
   Forward request to the web server
   Else
   Block the user
4. Generation of query by the web server.
5. Allocate separate session to each user.
6. Analyse query
If (malicious) then
   Reject query
Else
   Forward query to the database server
8. Give desired results to users.

VII. ATTACK SCENARIOS

Dual Sentinel is effective at capturing the following types of attacks:

A. Privilege Escalation Attack:

Let’s assume that the website serves both regular users and administrators. For a regular user, the web request $R_{user}$ will trigger the set of SQL queries $Q_{user}$ for an administrator, the request $R_{admin}$ will trigger the set of admin level queries $Q_{admin}$. Now suppose that an attacker logs into the web server as a normal user, improves his privileges, and activates admin queries so as to obtain an admin's data. This attack can never be noticed by the web server Intrusion Detection System or the database Intrusion Detection System since both $R_{user}$ and $Q_{admin}$ are valid requests and queries. Our approach, though, can detect this type of attack since the database query $Q_{admin}$ does not match the request $R_{user}$, according to our mapping model.

![Fig. 4: Privilege Escalation Attack](image)

B. Hijack Future Session Attack:

This category of attacks is mainly aimed at the web server side. The attacker generally takes over the web server and therefore hijacks all subsequent valid user sessions to start on attacks. For example, by hijacking other user sessions, the attacker is able to eavesdrop, send spoofed replies, and may also drop user requests. Fig.5 illustrates a scenario wherein a compromised web server can harm all the Hijack Future Sessions by not generating any database queries for normal-user or client requests.

According to the mapping model of Dual sentinel, the web request should appeal to some database queries (e.g., a Deterministic Mapping), then the uncharacteristic situation can be identified. Though, neither a predictable web server Intrusion Detection System nor a database Intrusion Detection System can detect such an attack by itself. Opportunely, the isolation property of our container based web server architecture can also prevent this type of attack. As each client’s web requests are being isolated into a separate container, an attacker will never able to break into other user’s sessions.

![Fig. 5: Hijack Future Session Attack](image)

C. Injection Attack

The SQL injection attacks do not require compromising the web server. Attackers can use existing susceptibilities in the web server logic to insert the data or string content that contains the exploits and then use the web server to communicate these exploits to assault the back end database. As our approach provides a two-tier detection, even though the exploits are accepted by web server, the transmitted contents to the database server would not be able to take on the expected structure for the given web server request. For example, since the SQL injection attack used to change the structure of the SQL queries, even though the injected data were to go through the web server side, it would produce SQL queries in a different structure that could be detected as a divergence from the SQL query structure and that would normally track such web requests.

![Fig. 6: Sql injection attack](image)
D. Direct DB Attack

It is possible for an attacker to sidestep the web server or also firewalls and connect directly to the database. An attacker could also have by now taken over the web server and can present such queries from the web server without sending the web requests. Without coordinated web requests for such queries, a web server Intrusion Detection System could identify neither. Additionally, if these DB queries were within the set of allowed queries, then the database Intrusion Detection System itself would not detect it either.

However, this type of attack can be caught with our approach since we cannot match any web requests with these queries. Fig. 7 illustrates the scenario wherein an attacker bypasses the webserver to directly query the database.

![Fig. 7: Direct DB Attack](image)

E. Cloning Attack

There exists different cases on which cloning attacks are detected, first one is when the cloned node uses any other present or current id as well as with same finger print. When the communication occurs between the network at that time clones are initiated and by cluster head if it is a cloned member node and base station if it is a cloned cluster head.

In the second case, when the cloned node uses same id with same finger print. At the initiation of the communication request, the base station can be detected immediately.

In the third case of cloning attack, the cloned node uses present/current id with a different finger print. Cluster heads could be detected when the finger print of the cloned node will not match with the finger print possessed by the cluster heads.

Base station has all the information about the nodes. By using zero knowledge protocol the cloning attacks are detected.

VIII. CONCLUSION AND FUTURE WORK

We propose an intrusion detection system i.e., Dual Sentinel, which constructs the model of normal behaviour for multitier web applications from both the front end web (HTTP) requests and back end DB (SQL) queries. Previous Intrusion Detection Systems interconnected or summarized alerts, whereas Dual Sentinel forms a container-based ID with multiple input streams to generate alerts. Such correlation of input streams provides a better characterization of the system for anomaly detection since the intrusion sensor has a more specific normality model that investigates a wider range of attacks. We achieved this by isolating the flow of information from each web server session with a lightweight virtualization.

Acknowledgement

We would like to thank Prof.S.A.Talekar for his guidance throughout our work and for his valuable comments and constant encouragement.

We would like to thank our HOD Prof. B.S.Tarle for providing necessary information and resources.

We would also like to convey our thank to our principal Prof. Dr.Jayant Pattiwar for providing all necessary facilities and his support.

REFERENCES


